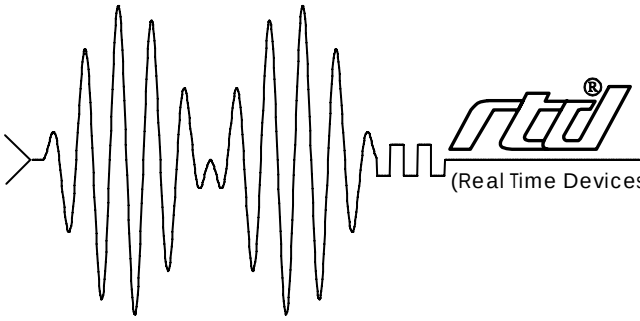
DM6604 / DM5604
User's Manual



RTD Embedded Technologies Inc.

"Accessing the Analog World"®

BDM-610010010
Rev. A

DM6604 / DM5604
User's Manual

RTD Embedded Technologies, INC.
103 Innovation Blvd.
State College, PA 16803-0906

Phone: +1-814-234-8087
FAX: +1-814-234-5218

E-mail
sales@rtd.com
techsupport@rtd.com

web site
http://www.rtd.com

Revision History

Rev. A New manual naming method

Published by:

RTD Embedded Technologies, Inc.
103 Innovation Blvd.
State College, PA 16803-0906

Copyright 1999, 2002, 2003 by RTD Embedded Technologies, Inc.
All rights reserved
Printed in U.S.A.

The RTD Logo is a registered trademark of RTD Embedded Technologies. cpuModule and utilityModule are trademarks of RTD Embedded Technologies. PhoenixPICO and PheonixPICO BIOS are trademarks of Phoenix Technologies Ltd. PS/2, PC/XT, PC/AT and IBM are trademarks of International Business Machines Inc. MS-DOS, Windows, Windows 95, Windows 98 and Windows NT are trademarks of Microsoft Corp. PC/104 is a registered trademark of PC/104 Consortium. All other trademarks appearing in this document are the property of their respective owners.

Table of Contents

INTRODUCTION ***i*-1**

Digital-to-Analog Conversion *i*-3
Digital I/O *i*-3
What Comes With Your Module *i*-3
Board Accessories *i*-3
AT Bus Connector P12 ..... *i*-4
Using This Manual *i*-4
When You Need Help *i*-4

CHAPTER 1 — MODULE SETTINGS **1-1**

Factory-Configured Switch and Jumper Settings 1-3
P3 Through P10 — DAC Output Voltage Range Select (Factory Setting: +5 to -5 volts, All Channels) 1-4
P11 — Interrupt Channel Select (Factory Setting: IRQ Disabled) 1-5
P13 — Interrupt Source Select (Factory Setting: G & EXT Connected) 1-5
S1 — Base Address (Factory Setting: 300 hex (768 decimal)) 1-6
Pull-up/Pull-down Resistors on Digital I/O Lines 1-7

CHAPTER 2 — BOARD INSTALLATION **2-1**

Module Installation 2-3
External I/O Connections 2-3
Connecting the Analog Outputs 2-4
Connecting the Digital I/O 2-4
Running the 5604DIAG Diagnostics Program 2-4

CHAPTER 3 — HARDWARE DESCRIPTION **3-1**

D/A Conversion 3-3
Digital I/O, Programmable Peripheral Interface 3-3
Interrupts 3-4

CHAPTER 4 — MODULE OPERATION AND PROGRAMMING **4-1**

Defining the I/O Map 4-3
BA + 0: PPI Port A — Digital I/O (Read/Write) 4-4
BA + 1: PPI Port B — Digital I/O (Read/Write) 4-4
BA + 2: PPI Port C — Digital I/O (Read/Write) 4-4
BA + 3: 8255 PPI Control Word (Write Only) 4-4
BA + 4: D/A Converter 1 LSB (Write Only) 4-6
BA + 5: D/A Converter 1 MSB (Write Only) 4-6
BA + 6: D/A Converter 2 LSB (Write Only) 4-6
BA + 7: D/A Converter 2 MSB (Write Only) 4-6
BA + 8: D/A Converter 3 LSB (Write Only) 4-6
BA + 9: D/A Converter 3 MSB (Write Only) 4-6
BA + 10: D/A Converter 4 LSB (Write Only) 4-6
BA + 11: D/A Converter 4 MSB (Write Only) 4-6
BA + 12: D/A Converter 5 LSB (Write Only) 4-6
BA + 13: D/A Converter 5 MSB (Write Only) 4-7
BA + 14: D/A Converter 6 LSB (Write Only) 4-7
BA + 15: D/A Converter 6 MSB (Write Only) 4-7

BA + 16: D/A Converter 7 LSB (Write Only) 4-7
BA + 17: D/A Converter 7 MSB (Write Only) 4-7
BA + 18: D/A Converter 8 LSB (Write Only) 4-7
BA + 19: D/A Converter 8 MSB (Write Only) 4-7
BA + 20: Update DAC Outputs (Write Only) 4-7
BA + 21: Reserved 4-7
BA + 22: IRQ Enable (Write Only) 4-7
BA + 23: Interrupt Status/Clear (Read/Write) 4-7
Programming the DM5604 4-8
Clearing and Setting Bits in a Port 4-9
D/A Conversions 4-10
Initializing the 8255 PPI 4-11
Digital I/O Operations 4-11
Interrupts 4-12
What Is an Interrupt? 4-12
Interrupt Request Lines 4-12
8259 Programmable Interrupt Controller 4-12
Interrupt Mask Register (IMR) 4-12
End-of-Interrupt (EOI) Command 4-12
What Exactly Happens When an Interrupt Occurs? 4-12
Using Interrupts in Your Programs 4-13
Writing an Interrupt Service Routine (ISR) 4-13
Saving the Startup Interrupt Mask Register (IMR) and Interrupt Vector 4-14
Restoring the Startup IMR and Interrupt Vector 4-15
Common Interrupt Mistakes 4-15
Example Programs 4-15
C Programs 4-15
BASIC Programs 4-15

CHAPTER 5 — CALIBRATION 5-1

Required Equipment 5-3
D/A Calibration — X2 Voltage Multiplier 5-3

APPENDIX A — DM5604 SPECIFICATIONS A-1

APPENDIX B — P2 CONNECTOR PIN ASSIGNMENTS B-1

APPENDIX C — COMPONENT DATA SHEETS C-1

APPENDIX D — WARRANTY D-1

List of Illustrations

1-1 Module Layout Showing Factory-Configured Settings 1-4
1-2 DAC Output Voltage Range Jumpers, P3 Through P10 1-4
1-3 Interrupt Channel Select Jumper, P11 1-5
1-4 Interrupt Source Select Jumpers, P13 1-5
1-5 Pulling Down the Interrupt Request Line 1-6
1-6 Base Address Switch, S1 1-7
1-7 Pull-up/Pull-down Resistor Circuitry 1-8
1-8 Adding Pull-ups and Pull-downs to Some Digital I/O Lines 1-8
2-1 P2 I/O Connector Pin Assignments 2-4
2-2 Voltage Output Connections 2-5
3-1 DM5604 Block Diagram 3-3
5-1 Module Layout 5-3

INTRODUCTION

The DM5604 dataModule® analog output board turns your IBM PC-compatible cpuModule™ or other PC/104 computer into a high-performance testing and control system. Ultra-compact for embedded and portable applications, the DM5604 features:

- Eight fast-settling 12-bit analog output channels,
- ±5, ±10, 0 to +5, or 0 to +10 volt analog output range,
- Simultaneous updating of all output channels,
- 24 TTL/CMOS 8255 based programmable digital I/O lines,
- Software enabled interrupts IRQ2-IRQ7, IRQ9-IRQ15 (DM6604)
- +5 volt only operation, 2.35W power consumption,
- BASIC and Turbo C source code; diagnostics program.

The following paragraphs briefly describe the major functions of the module. More detailed discussions of module functions are included in Chapter 3, *Hardware Description*, and Chapter 4, *Module Operation and Programming*. The module setup is described in Chapter 1, *Module Settings*.

Digital-to-Analog Conversion

The digital-to-analog (D/A) circuitry features two 12-bit converter channels in each AD7237 D/A converter IC for a total of eight output channels. The two channels in each AD7237 are internally double buffered and all channels are simultaneously updated by issuing a single command. Each channel can be jumpered to one of four output voltage ranges, ±5, ±10, 0 to +5, or 0 to +10 volts.

Digital I/O

The DM5604 has 24 TTL/CMOS-compatible digital I/O lines which can be directly interfaced with external devices or signals to sense switch closures, trigger digital events, or activate solid-state relays. These lines are provided by the on-board 8255 programmable peripheral interface chip. The 8255 can be operated in one of three modes. Pads for installing and activating pull-up or pull-down resistors are included on the module. Installation procedures are given at the end of Chapter 1, *Module Settings*.

What Comes With Your Module

You receive the following items in your DM5604 package:

- DM5604 interface module with stackthrough bus header
- Software and diagnostics diskette with BASIC, Turbo Pascal, and Turbo C source code
- User's manual

If any item is missing or damaged, please call Real Time Devices' Customer Service Department at (814) 234-8087. If you require service outside the U.S., contact your local distributor.

Board Accessories

In addition to the items included in your DM5604 package, Real Time Devices offers a full line of accessories. Call your local distributor or our main office for more information about these accessories and for help in choosing the best items to support your board's application.

Accessories for the DM5604 include the MR8/MR16 series 8 or 16 channel mechanical relay boards, the OP8/OP16 series 8 or 16 channel optoisolated digital input boards, the OR16 mechanical relay/optoisolated digital I/O board, the TB50 terminal board and XB50 prototype/terminal board for prototype development and easy signal access, the DM14 extender board for testing your module in a PC compatible computer, and the XT50 twisted pair wire flat ribbon cable assembly for external interfacing.

AT Bus Connector P12 (DM6604)

The DM6604 is exactly the same as the DM5604 except for the addition of the AT bus connector P12. This allows you to stack the module with CPU's that have the AT bus connectors and access the AT interrupts.

Using This Manual

This manual is intended to help you install your new module and get it running quickly, while also providing enough detail about its functions so that you can enjoy maximum use of its features even in the most complex applications. We assume that you already have an understanding of data acquisition principles and that you can customize the example software or write your own applications programs.

When You Need Help

This manual and the example programs in the software package included with your module provide enough information to properly use all of the module's features. If you have any problems installing or using this module, contact our Technical Support Department, (814) 234-8087, during regular business hours, eastern standard time or eastern daylight time, or send a FAX requesting assistance to (814) 234-5218. When sending a FAX request, please include your company's name and address, your name, your telephone number, and a brief description of the problem.

CHAPTER 1

MODULE SETTINGS

The DM5604 has jumper and switch settings you can change if necessary for your application. The module is factory-configured as listed in the table and shown on a diagram in the beginning of this chapter. Should you need to change these settings, use these easy-to-follow instructions before you install the module in your PC/104 system.

Note that by installing resistor packs at four the RN locations around the 8255 PPI and soldering a jumper between +5V and common or ground and common in the associated pads for each resistor network, you can configure groups of digital I/O lines to be pulled up or pulled down. This procedure is explained at the end of this chapter.

Factory-Configured Switch and Jumper Settings

Table 1-1 lists the factory settings of the user-configurable jumper and switches on the DM5604. Figure 1-1 shows the module layout and the locations of the factory-set jumpers. The following paragraphs explain how to change the factory settings. Pay special attention to the setting of S1, the base address switch, to avoid address contention when you first use your board in your system.

Table 1-1 Factory Settings

Switch/ Jumper	Function Controlled	Factory Settings (Jumpers Installed)
P3	Configures the output voltage range for DAC1	Jumpers installed on ±5 & X1 to set output of -5 to +5 volts
P4	Configures the output voltage range for DAC2	Jumpers installed on ±5 & X1 to set output of -5 to +5 volts
P5	Configures the output voltage range for DAC3	Jumpers installed on ±5 & X1 to set output of -5 to +5 volts
P6	Configures the output voltage range for DAC4	Jumpers installed on ±5 & X1 to set output of -5 to +5 volts
P7	Configures the output voltage range for DAC5	Jumpers installed on ±5 & X1 to set output of -5 to +5 volts
P8	Configures the output voltage range for DAC6	Jumpers installed on ±5 & X1 to set output of -5 to +5 volts
P9	Configures the output voltage range for DAC7	Jumpers installed on ±5 & X1 to set output of -5 to +5 volts
P10	Configures the output voltage range for DAC8	Jumpers installed on ±5 & X1 to set output of -5 to +5 volts
P11	Selects an interrupt channel	Interrupt channels disabled
P13	Connects one of three interrupt sources to an interrupt channel; pulls tri-state buffer to ground (G) for multiple interrupt applications	Jumpers installed on on G (ground for buffer) & EXT;
S1	Sets the base address	300 hex (768 decimal)

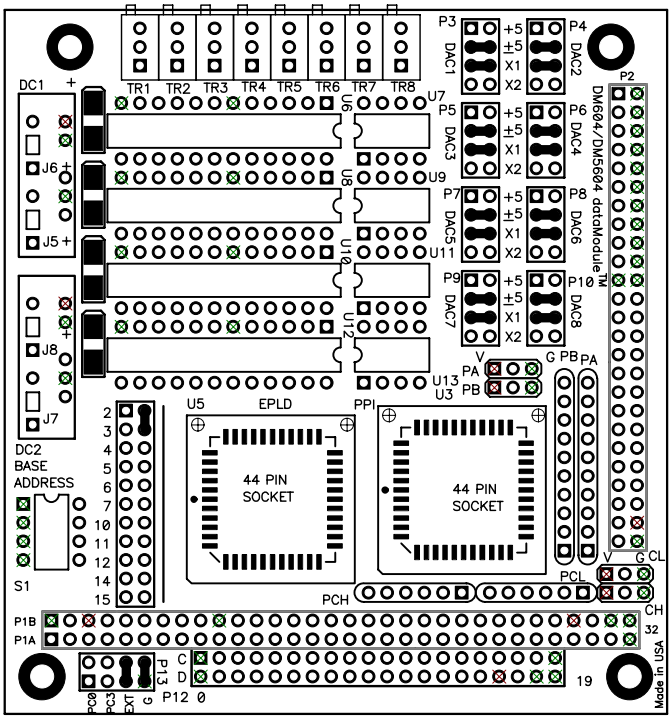


Fig. 1-1 — Module Layout Showing Factory-Configured Settings

P3 Through P10 — DAC Output Voltage Range Select (Factory Setting: +5 to -5 volts, All Channels)

These eight header connectors individually set the output voltage range for the eight DAC channels at 0 to +5, ±5, 0 to +10, or ±10 volts. Two jumpers must be installed on each header, one to select the range and one to select the multiplier. The top two jumpers select the range, bipolar (±5) or unipolar (5). The bottom two jumpers select the multiplier, X2 or X1. When a jumper is on the X2 multiplier pins, the range values become ±10 and 10. The table below shows the four possible combinations of jumper settings for each DAC channel, and Figure 1-2 shows the factory setting for DAC1 (P3). NOTE: At power up or CPU reset, the analog outputs are set to 0 volts. When the first DAC update command is issued all DAC outputs will be switched on.

Jumpers (top to bottom)	Output Range			
	±5V	0 to +5V	±10V	0 to +10V
5V	OFF	ON	OFF	ON
±5V	ON	OFF	ON	OFF
X1	ON	ON	OFF	OFF
X2	OFF	OFF	ON	ON

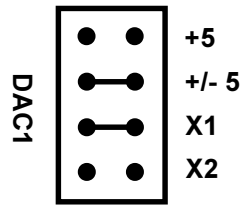


Fig. 1-2 — DAC Output Voltage Range Jumpers, P3 Through P10

P11 — Interrupt Channel Select (Factory Setting: Disabled)

This header connector, shown in Figure 1-3, lets you connect one of three interrupt sources to an interrupt channel. To connect the interrupt source to an interrupt channel, you must install one jumper across the desired source on P13 and a second jumper across the desired IRQ channel on P11. The available channels are: IRQ2 - IRQ7, IRQ10, IRQ11, IRQ12, IRQ14 & IRQ15. Note that to use IRQ10 - IRQ15, you must be using an AT CPU module that supports these interrupts and you must also have P12 installed on your DM6604 module. Please contact the factory for more information on this option.

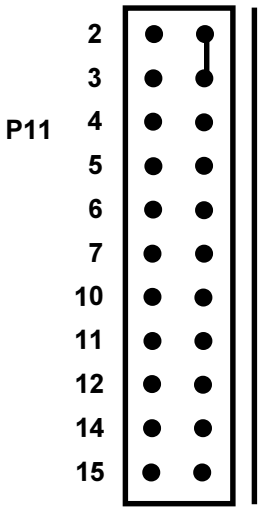


Fig. 1-3 — Interrupt Channel Select Jumper, P11

P13 — Interrupt Source Select (Factory Setting: G & EXT Connected)

This header connector, shown in Figure 1-4, lets you connect one of three interrupt sources to an interrupt channel. To connect the interrupt source to an interrupt channel, you must install one jumper across the desired source and a second jumper across the desired IRQ channel on P11. The available sources are: PC0, which is the INTRB signal from the 8255 PPI; PC3, which is the INTRA signal from the 8255 PPI; and EXT, an external interrupt you can route onto the board through the P2 I/O connector. Note that only ONE interrupt source can be activated at a time.

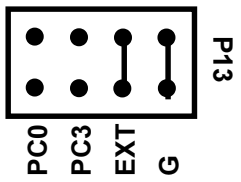


Fig. 1-4 — Interrupt Source Select Jumpers, P13

This module supports an interrupt sharing mode where the pins on P13 labeled G are provided so that you can install a jumper which connects a 1 kilohm pull-down resistor to the output of a high-impedance tri-state driver which carries the interrupt request signal. This pull-down resistor pulls the interrupt request line low whenever interrupts are not active. So, whenever an interrupt request is made, the tri-state buffer is enabled, forcing the output high and causing an interrupt. You can monitor the interrupt status through bit 0 in the status word (I/O address location BA + 23). After the interrupt has been serviced, the clear command returns the IRQ line low, disabling the tri-state buffers, and pulling the output low again. Figure 1-4 shows this circuit. Because the interrupt request line is pulled low only by the pull-down resistor, you can have two or more modules which share

the same IRQ channel. You can tell which module issued the interrupt request by monitoring each module's IRQ status bit. If you are not planning on sharing interrupts or if you are not sure that your CPU supports interrupt sharing, it is best to disable this feature and use the interrupts in the normal mode. This will insure compatibility with all CPUs. See chapter 4 for details on disabling the interrupt sharing circuit.

NOTE: When you use multiple modules that share the same interrupt, only one module should have the G ground jumper installed. The rest should be disconnected. Whenever you operate a single module, the G jumper should be installed. Whenever you operate the module with interrupt sharing disabled, the G jumper should be removed.

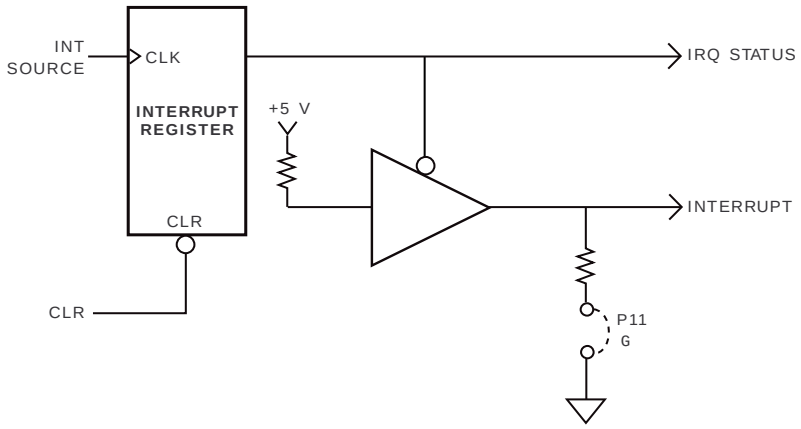


Fig. 1-5 — Pulling Down the Interrupt Request Line

S1 — Base Address (Factory Setting: 300 hex (768 decimal))

One of the most common causes of failure when you are first trying your module is address contention. Some of your computer's I/O space is already occupied by internal I/O and other peripherals. When the DM5604 attempts to use I/O address locations already used by another device, contention results and the module does not work.

To avoid this problem, the DM5604 has an easily accessible DIP switch, S1, which lets you select any one of 16 starting addresses in the computer's I/O. Should the factory setting of 300 hex (768 decimal) be unsuitable for your system, you can select a different base address simply by setting the switches to any value shown in Table 1-2. The table shows the switch settings and their corresponding decimal and hexadecimal (in parentheses) values. Make sure that you verify the order of the switch numbers on the switch (1 through 4) before setting them. When the switches are pulled forward, they are OPEN, or set to logic 1, as labeled on the DIP switch package. When you set the base address for your board, record the value in the table inside the back cover. Figure 1-6 shows the DIP switch set for a base address of 300 hex (768 decimal).

Table 1-2 Base Address Switch Settings, S1			
Base Address Decimal / (Hex)	**Switch Setting 4 3 2 1**	**Base Address Decimal / (Hex)**	**Switch Setting 4 3 2 1**
512 / (200)	0 0 0 0	768 / (300)	1 0 0 0
544 / (220)	0 0 0 1	800 / (320)	1 0 0 1
576 / (240)	0 0 1 0	832 / (340)	1 0 1 0
608 / (260)	0 0 1 1	864 / (360)	1 0 1 1
640 / (280)	0 1 0 0	896 / (380)	1 1 0 0
672 / (2A0)	0 1 0 1	928 / (3A0)	1 1 0 1
704 / (2C0)	0 1 1 0	960 / (3C0)	1 1 1 0
736 / (2E0)	0 1 1 1	992 / (3E0)	1 1 1 1
0 = closed, 1 = open			

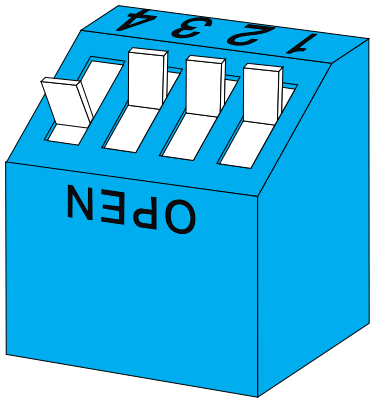


Fig. 1-6 — Base Address Switch, S1

Pull-up/Pull-down Resistors on Digital I/O Lines

The 8255 programmable peripheral interface provides 24 parallel TTL/CMOS compatible digital I/O lines which can be interfaced with external devices. The lines are divided into four groups: eight Port A lines, eight Port B lines, four Port C Lower lines, and four Port C Upper lines. You can install and connect pull-up or pull-down resistors for any or all of these four groups of lines. You may want to pull lines up for connection to switches. This will pull the line high when the switch is disconnected. Or, you may want to pull down lines connected to relays which control turning motors on and off. These motors turn on when the digital lines controlling them are high. Pulling these lines down keeps them from floating high during the brief period between power-up and initialization.

To use the pull-up/pull-down feature, you must first install resistor packs in any or all of the four locations around the 8255, labeled PA, PB, PCL, and PCH. PA (RN3) and PB (RN6) take a 10-pin pack, and PCL (RN4) and PCH (RN5) take 6-pin packs. Figure 1-7 shows these locations.

After the resistor packs are installed, you must connect them into the circuit as pull-ups or pull-downs. Locate the three-hole pads on the board near the resistor packs. They are labeled G (for ground) on one end and V (for +5 volts) on the other end. The middle hole is common. PA is for Port A, PB for Port B, PCL is for Port C Lower, and PCH is for Port C Upper. Figure 1-7 shows a blowup of the pads. To operate as pull-ups, solder a jumper wire between the common pin (middle pin of the three) and the V pin. For pull-downs, solder a jumper wire between the common pin (middle pin) and the G pin. For example, Figure 1-8 shows Port A lines with pull-ups, Port C Lower with pull-downs, and Port C Upper with no resistors.

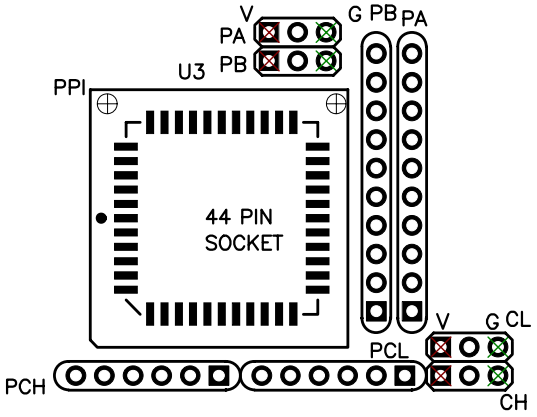


Fig. 1-7 — Pull-up/Pull-down Resistor Circuitry

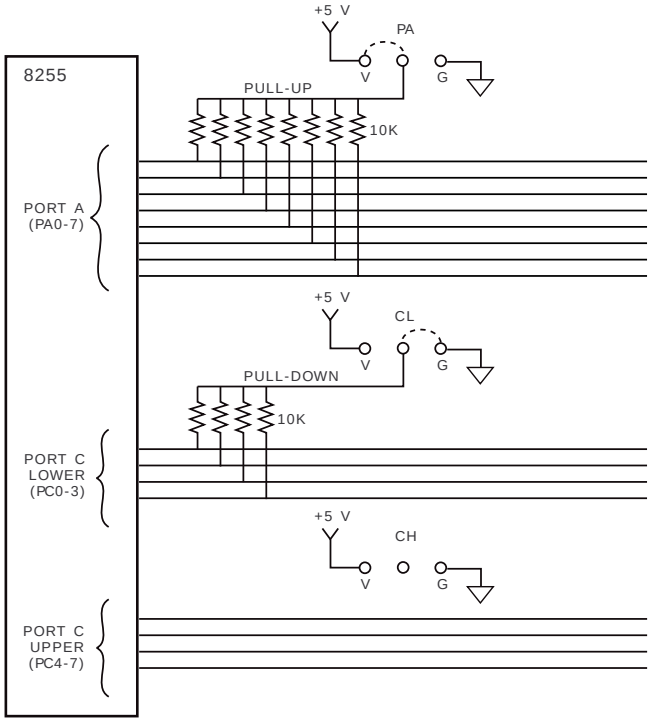


Fig. 1-8 — Adding Pull-ups and Pull-downs to Some Digital I/O Lines

CHAPTER 2

BOARD INSTALLATION

The DM5604 is easy to install in your cpuModule™ or other PC/104 based system. This chapter tells you step-by-step how to install and connect the module.

After you have installed the module and made all of your connections, you can turn your system on and run the 5604DIAG board diagnostics program included on your example software disk to verify that your module is working.

Module Installation

Keep the module in its antistatic bag until you are ready to install it in your cpuModule™ or other PC/104 based system. When removing it from the bag, hold the module at the edges and do not touch the components or connectors.

Before installing the module in your system, check the jumper and switch settings. Chapter 1 reviews the factory settings and how to change them. If you need to change any settings, refer to the appropriate instructions in Chapter 1. Note that incompatible jumper settings can result in unpredictable operation and erratic response.

The DM5604 comes with a stackthrough P1 connector. The stackthrough connector lets you stack another board on top of your DM5604, plugging it into the data bus through the pins on the non-component side of the board.

To install the module, follow the procedures described in the computer manual and the steps below:

1. Turn OFF the power to your system.
2. Touch a metal frame or to discharge any static buildup and then remove the module from its antistatic bag.
3. Select the appropriate standoffs for your application to secure the board when you install it in your system (two sizes are included with the board).
4. Holding the module by its edges, orient it so that the P1 bus connector's pin 1 lines up with pin 1 of the expansion connector onto which you are installing the module.
5. After carefully positioning the module so that the card edge connector is resting on the expansion connector, gently and evenly press down until it is secured on the connector.

   NOTE: Do not force the module onto the connector. If the module does not slide into place, remove it and try again. Wiggling the module or exerting too much pressure can result in damage to the DM5604 or to the computer board.
6. After the module is installed, connect the cable to I/O connector P2 on the module. When making this connection, note that there is no keying to guide you in orientation. You must make sure that pin 1 of the cable is connected to pin 1 of P2 (pin 1 is marked on the module with a small square). For twisted pair cables, pin 1 is the dark brown wire; for standard single wire cables, pin 1 is the red wire.
7. Make sure all connections are secure.

External I/O Connections

Figure 2-1 shows the DM5604's P2 I/O connector pinout. Refer to this diagram as you make your I/O connections.

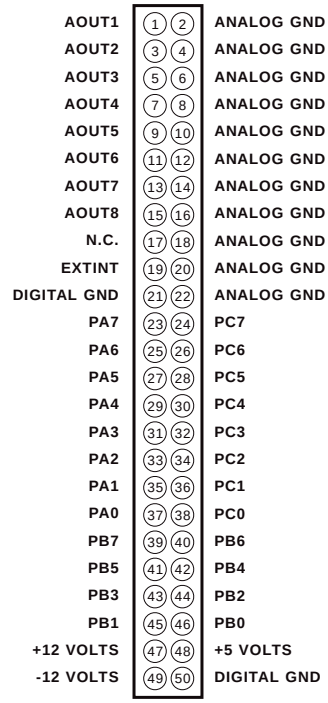

AOUT1 1 2 ANALOG GND
AOUT2 3 4 ANALOG GND
AOUT3 5 6 ANALOG GND
AOUT4 7 8 ANALOG GND
AOUT5 9 10 ANALOG GND
AOUT6 11 12 ANALOG GND
AOUT7 13 14 ANALOG GND
AOUT8 15 16 ANALOG GND
N.C. 17 18 ANALOG GND
EXTINT 19 20 ANALOG GND
DIGITAL GND 21 22 ANALOG GND
PA7 23 24 PC7
PA6 25 26 PC6
PA5 27 28 PC5
PA4 29 30 PC4
PA3 31 32 PC3
PA2 33 34 PC2
PA1 35 36 PC1
PA0 37 38 PC0
PB7 39 40 PB6
PB5 41 42 PB4
PB3 43 44 PB2
PB1 45 46 PB0
+12 VOLTS 47 48 +5 VOLTS
-12 VOLTS 49 50 DIGITAL GND

Fig. 2-1 — P2 I/O Connector Pin Assignments

Connecting the Analog Outputs

For each of the eight output channels, the high side of the device receiving the output is connected to an AOUT line and the low side is connected to the corresponding ANALOG GND. Figure 2-2 shows how to connect the DM5604 voltage outputs to a load.

> **NOTE:** Driving highly capacitive loads can cause unwanted oscillations. Placing a 100 ohm resistor in series with the capacitive load should eliminate this condition.

Connecting the Digital I/O

For all of the digital connections, the high side of an external signal source or destination device is connected to the appropriate signal pin on the I/O connector, and the low side is connected to any DIGITAL GND.

Running the 5604DIAG Diagnostics Program

Now that your board is ready to use, you will want to try it out. An easy-to-use, menu-driven diagnostics program, 5604DIAG, is included with your example software to help you verify your board's operation. You can also use this program to make sure that your current base address setting does not contend with another device.

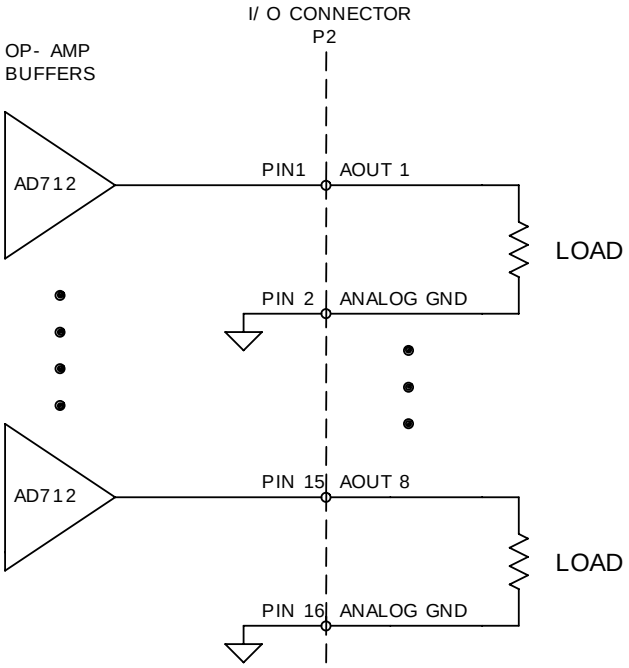


Fig. 2-2 — Voltage Output Connections

CHAPTER 3

HARDWARE DESCRIPTION

This chapter describes the major features of the DM5604. This chapter also describes the hardware-selectable interrupts.

The DM5604 provides eight analog output channels with selectable voltage output ranges and 24 TTL/CMOS digital I/O lines, as shown Figure 3-1. This chapter describes the hardware which makes up the major functions and hardware-selectable interrupts.

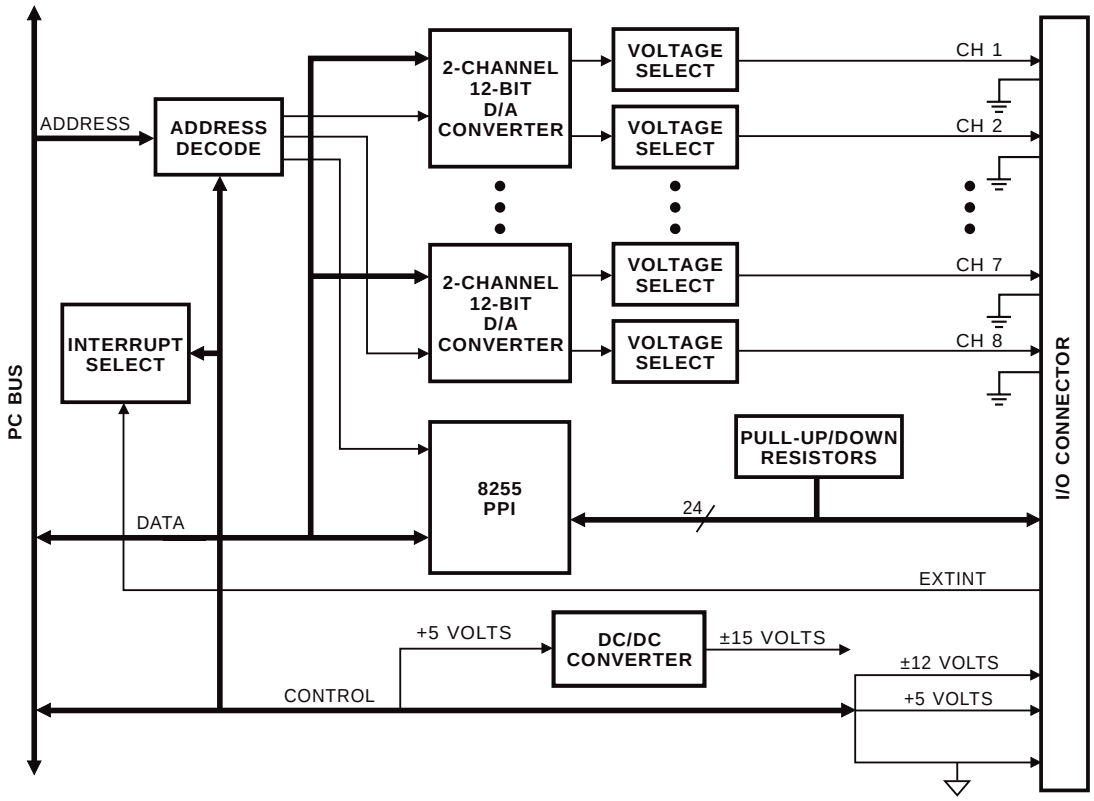


Fig. 3-1 — DM5604 Block Diagram

D/A Conversion

The DM5604 board performs digital-to-analog conversions on eight independent analog output channels. The output of each conversion channel is jumper-selectable for ±5, ±10, 0 to +5, or 0 to +10 volts.

The AD7237 12-bit D/A converter contains two independent D/A converter channels in a single CMOS package. The data to be converted is double buffered at the D/A input, which allows simultaneous updating of all four D/A output channels.

The AD712 precision operational amplifier provides complete coverage of the output voltage ranges. The exceptionally low offset voltage and drift ensure an accurate analog output on each channel.

The power-up condition of the D/A outputs is 0 volts. This is accomplished by an analog switch on the output of each D/A channel. At power-up or at CPU reset, this switch is connected to ground. When the first update command is issued to the D/A converters, this switch switches from ground to the output of the D/A's. When you are first initializing the board, it is best to program each D/A before sending the update command.

Digital I/O, Programmable Peripheral Interface

The programmable peripheral interface (PPI) is used for digital I/O functions. This high-performance TTL/CMOS compatible chip has 24 digital I/O lines divided into two groups of 12 lines each:

Group A — Port A (8 lines) and Port C Upper (4 lines);
Group B — Port B (8 lines) and Port C Lower (4 lines).

You can program these ports for one of three modes of operation:

Mode 0 — Basic input/output. Lets you use simple input and output operation for a port. Data is written to or read from the specified port.

Mode 1 — Strobed input/output. Lets you transfer I/O data from Port A in conjunction with strobes or handshaking signals.

Mode 2 — Strobed bidirectional input/output. Lets you communicate bidirectionally with an external device through Port A. Handshaking is similar to Mode 1.

These modes are detailed in the 8255 Data Sheet, reprinted from Intel in Appendix C.

Interrupts

The DM5604 has three jumper-selectable interrupt sources: an external trigger (EXT), the INTRB (PC0) interrupt generated by the 8255 PPI in Mode 1, and the INTRA interrupt (PC3) generated by the 8255 PPI . Chapter 1 tells you how to set the jumpers on the interrupt header connectors, P11 & P12, and Chapter 4 describes how to program interrupts.

CHAPTER 4

MODULE OPERATION AND PROGRAMMING

This chapter shows you how to program and use your DM5604. It provides a complete description of the I/O map and programming operations to aid you in programming. The example programs included on the disk in your module package are listed at the end of this chapter. These programs, written in Turbo C and BASIC, include source code to simplify your applications programming.

Defining the I/O Map

The I/O map for the DM5604 is shown in Table 4-1 below. As shown, the board occupies 24 consecutive I/O port locations. The base address (designated as BA) can be selected using DIP switch S1 as described in Chapter 1, *Board Settings*. This switch can be accessed without removing the board from the stack. The following sections describe the register contents of each address used in the I/O map.

Table 4-1 DM5604 I/O Map

Register Description	Read Function	Write Function	Address * (Decimal)
8255 PPI Port A	Read Port A digital input lines	Program Port A digital output lines	BA + 0
8255 PPI Port B	Read Port B digital input lines	Program Port B digital output lines	BA + 1
8255 PPI Port C	Read Port C digital input lines	Program Port C digital output lines	BA + 2
8255 PPI Control Word	Reserved	Program PPI configuration	BA + 3
D/A Converter 1 LSB	Reserved	Program DAC1 LSB	BA + 4
D/A Converter 1 MSB	Reserved	Program DAC1 MSB	BA + 5
D/A Converter 2 LSB	Reserved	Program DAC2 LSB	BA + 6
D/A Converter 2 MSB	Reserved	Program DAC2 MSB	BA + 7
D/A Converter 3 LSB	Reserved	Program DAC3 LSB	BA + 8
D/A Converter 3 MSB	Reserved	Program DAC3 MSB	BA + 9
D/A Converter 4 LSB	Reserved	Program DAC4 LSB	BA + 10
D/A Converter 4 MSB	Reserved	Program DAC4 MSB	BA + 11
D/A Converter 5 LSB	Reserved	Program DAC5 LSB	BA + 12
D/A Converter 5 MSB	Reserved	Program DAC5 MSB	BA + 13
D/A Converter 6 LSB	Reserved	Program DAC6 LSB	BA + 14
D/A Converter 6 MSB	Reserved	Program DAC6 MSB	BA + 15
D/A Converter 7 LSB	Reserved	Program DAC7 LSB	BA + 16
D/A Converter 7 MSB	Reserved	Program DAC7 MSB	BA + 17
D/A Converter 8 LSB	Reserved	Program DAC8 LSB	BA + 18
D/A Converter 8 MSB	Reserved	Program DAC8 MSB	BA + 19
Update All DACs	Reserved	Updates the outputs of all DACs	BA + 20
Reserved	Reserved	Reserved	BA + 21
IRQ Enable	Reserved	Enable interrupt generation/ Disable interrupt sharing	BA + 22
Interrupt Status/Clear	Read status of interrupt	Clear interrupt	BA + 23
* BA = Base Address			

BA + 0: PPI Port A — Digital I/O (Read/Write)

Transfers the 8-bit Port A digital input and digital output data between the board and an external device. A read transfers data from the external device, through P2, and into PPI Port A; a write transfers the written data from Port A through P2 to an external device.

BA + 1: PPI Port B — Digital I/O (Read/Write)

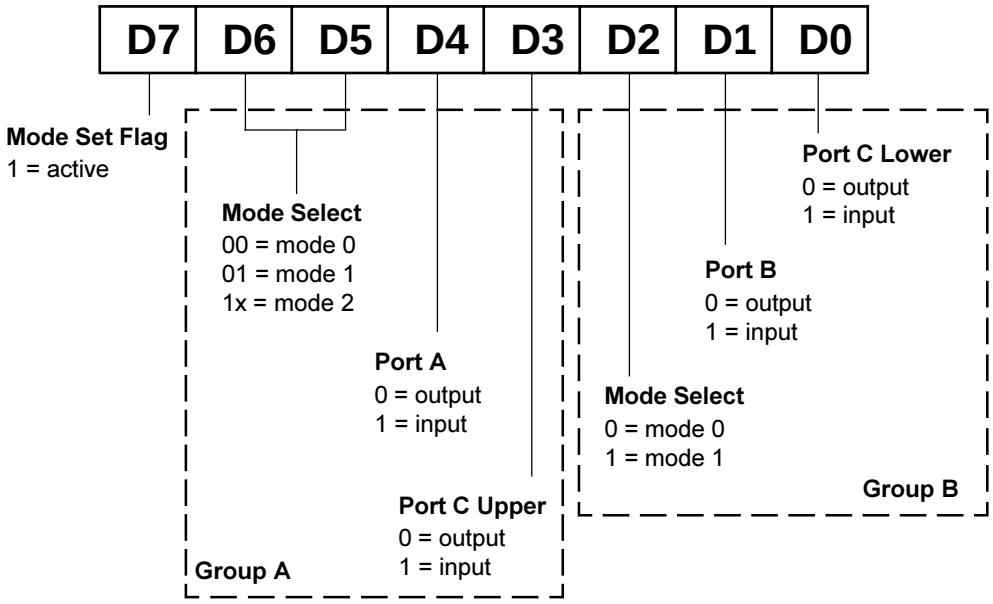
Transfers the 8-bit Port B digital input and digital output data between the board and an external device. A read transfers data from the external device, through on-board I/O connector P2, and into PPI Port B; a write transfers the written data from Port B through P2 to an external device.

BA + 2: PPI Port C — Digital I/O (Read/Write)

Transfers the two 4-bit Port C digital input and digital output data groups (Port C Upper and Port C Lower) between the board and an external device. A read transfers data from the external device, through P2, and into PPI Port C; a write transfers the written data from Port C through P2 to an external device.

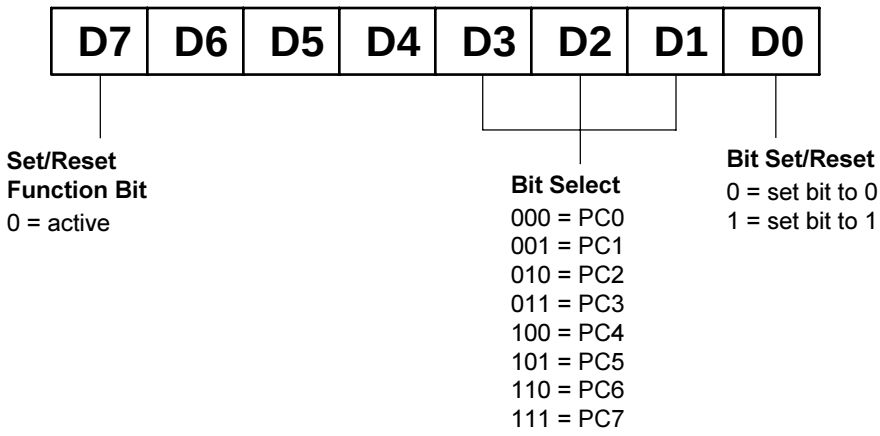
BA + 3: 8255 PPI Control Word (Write Only)

When bit 7 of this word is set to 1, a write programs the PPI configuration. The table at the top of the next page shows the control words for the 16 possible Mode 0 Port I/O combinations.

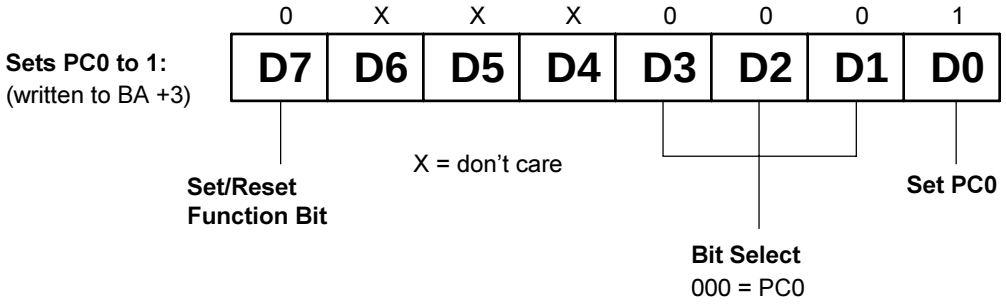


8255 Port I/O Flow Direction and Control Words, Mode 0						
Group A		Group B		Control Word		
Port A	Port C Upper	Port B	Port C Lower	Binary	Decimal	Hex
Output	Output	Output	Output	1 0 0 0 0 0 0 0	128	80
Output	Output	Output	Input	1 0 0 0 0 0 0 1	129	81
Output	Output	Input	Output	1 0 0 0 0 0 1 0	130	82
Output	Output	Input	Input	1 0 0 0 0 0 1 1	131	83
Output	Input	Output	Output	1 0 0 0 1 0 0 0	136	88
Output	Input	Output	Input	1 0 0 0 1 0 0 1	137	89
Output	Input	Input	Output	1 0 0 0 1 0 1 0	138	8A
Output	Input	Input	Input	1 0 0 0 1 0 1 1	139	8B
Input	Output	Output	Output	1 0 0 1 0 0 0 0	144	90
Input	Output	Output	Input	1 0 0 1 0 0 0 1	145	91
Input	Output	Input	Output	1 0 0 1 0 0 1 0	146	92
Input	Output	Input	Input	1 0 0 1 0 0 1 1	147	93
Input	Input	Output	Output	1 0 0 1 1 0 0 0	152	98
Input	Input	Output	Input	1 0 0 1 1 0 0 1	153	99
Input	Input	Input	Output	1 0 0 1 1 0 1 0	154	9A
Input	Input	Input	Input	1 0 0 1 1 0 1 1	155	9B

When bit 7 of this word is set to 0, a write can be used to individually program the Port C lines.



For example, if you want to set Port C bit 0 to 1, you would set up the control word so that bit 7 is 0; bits 1, 2, and 3 are 0 (this selects PC0); and bit 0 is 1 (this sets PC0 to 1). The control word is set up like this:

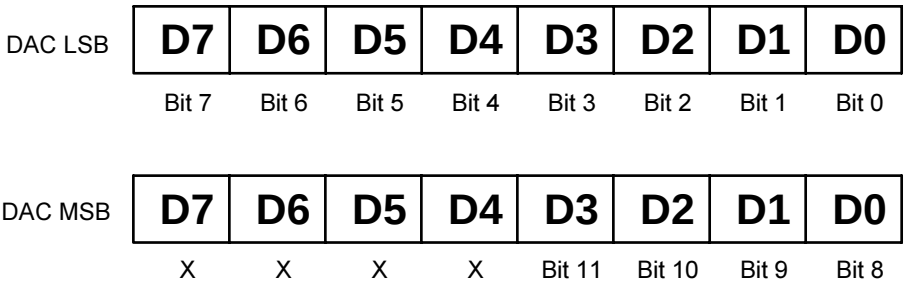


BA + 4: D/A Converter 1 LSB (Write Only)

Programs the DAC1 LSB (eight bits).

BA + 5: D/A Converter 1 MSB (Write Only)

Programs the DAC1 MSB (four bits) into D0 through D3; D4 through D7 are irrelevant.



BA + 6: D/A Converter 2 LSB (Write Only)

Programs the DAC2 LSB (eight bits).

BA + 7: D/A Converter 2 MSB (Write Only)

Programs the DAC2 MSB (four bits) into D0 through D3; D4 through D7 are irrelevant.

BA + 8: D/A Converter 3 LSB (Write Only)

Programs the DAC3 LSB (eight bits).

BA + 9: D/A Converter 3 MSB (Write Only)

Programs the DAC3 MSB (four bits) into D0 through D3; D4 through D7 are irrelevant.

BA + 10: D/A Converter 4 LSB (Write Only)

Programs the DAC4 LSB (eight bits).

BA + 11: D/A Converter 4 MSB (Write Only)

Programs the DAC4 MSB (four bits) into D0 through D3; D4 through D7 are irrelevant.

BA + 12: D/A Converter 5 LSB (Write Only)

Programs the DAC5 LSB (eight bits).

BA + 13: D/A Converter 5 MSB (Write Only)

Programs the DAC5 MSB (four bits) into D0 through D3; D4 through D7 are irrelevant.

BA + 14: D/A Converter 6 LSB (Write Only)

Programs the DAC6 LSB (eight bits).

BA + 15: D/A Converter 6 MSB (Write Only)

Programs the DAC6 MSB (four bits) into D0 through D3; D4 through D7 are irrelevant.

BA + 16: D/A Converter 7 LSB (Write Only)

Programs the DAC7 LSB (eight bits).

BA + 17: D/A Converter 7 MSB (Write Only)

Programs the DAC7 MSB (four bits) into D0 through D3; D4 through D7 are irrelevant.

BA + 18: D/A Converter 8 LSB (Write Only)

Programs the DAC8 LSB (eight bits).

BA + 19: D/A Converter 8 MSB (Write Only)

Programs the DAC8 MSB (four bits) into D0 through D3; D4 through D7 are irrelevant.

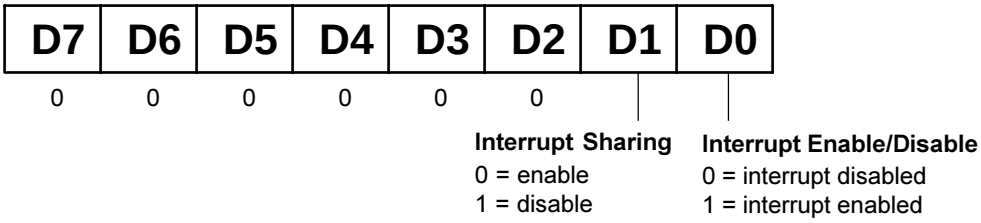
BA + 20: Update DAC Outputs (Write Only)

A write simultaneously starts a D/A conversion in all eight channels (data written is irrelevant). If the data has not been updated since the last conversion, the output of the DAC will not change.

BA + 21: Reserved

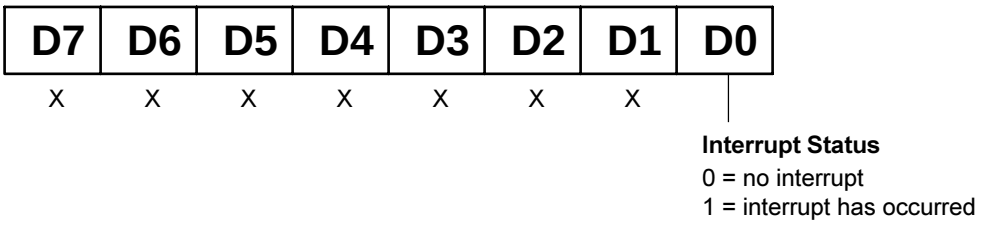
BA + 22: IRQ Enable (Write Only)

A write enables and disables interrupt generation and interrupt sharing. Writing a “1” to bit 0 enables interrupt generation; writing a “0” disables interrupt generation, writing a “1” to bit 1 disables interrupt sharing;writing a “0” enables interrupt sharing. At power up this register is set to "0".


D7 D6 D5 D4 D3 D2 D1 D0
0 0 0 0 0 0
Interrupt Sharing
0 = enable
1 = disable
Interrupt Enable/Disable
0 = interrupt disabled
1 = interrupt enabled

BA + 23: Interrupt Status/Clear (Read/Write)

A read shows the status of the interrupt (bit 0 only) as defined below. A write clears the interrupt (data written is irrelevant). Each time the interrupt status bit goes high, a write should follow to clear the bit.



Programming the DM5604

This section gives you some general information about programming and the DM5604, and then walks you through the major DM5604 programming functions. These descriptions will help you as you use the example programs included with the module. All of the program descriptions in this section use decimal values unless otherwise specified.

The DM5604 is programmed by writing to and reading from the correct I/O port locations on the board. These I/O ports were defined in the previous section. Most high-level languages such as BASIC, Pascal, C, and C++, and of course assembly language, make it very easy to read/write these ports. The table below shows you how to read from and write to I/O ports using some popular programming languages.

Language	Read	Write
BASIC	Data=INP(Address)	OUT Address,Data
Turbo C	Data=inportb(Address)	outportb(Address,Data)
Turbo Pascal	Data:=Port[Address]	Port[Address]:=Data

Assembly	mov dx,Address
	in al,dx
		mov dx,Address
		mov al,Data
		out dx,al

In addition to being able to read/write the I/O ports on the DM5604, you must be able to perform a variety of operations that you might not normally use in your programming. The table below shows you some of the operators discussed in this section, with an example of how each is used with Pascal, C, and BASIC. Note that the modulus operator is used to retrieve the least significant byte (LSB) of a two-byte word, and the integer division operator is used to retrieve the most significant byte (MSB).

Language	Modulus	Integer Division	AND	OR
C	% a = b % c	/ a = b / c	& a = b & c	| a = b | c
Pascal	MOD a := b MOD c	DIV a := b DIV c	AND a := b AND c	OR a := b OR c
BASIC	MOD a = b MOD c	\ a = b \ c	AND a = b AND c	OR a = b OR c

Many compilers have functions that can read/write either 8 or 16 bits from/to an I/O port. For example, Turbo Pascal uses **Port** for 8-bit port operations and **PortW** for 16 bits, Turbo C uses **inportb** for an 8-bit read of a port and **inport** for a 16-bit read. **Be sure to use only 8-bit operations with both the DM5604 and the DM6604!**

Clearing and Setting Bits in a Port

When you clear or set one or more bits in a port, you must be careful that you do not change the status of the other bits. You can preserve the status of all bits you do not wish to change by proper use of the AND and OR binary operators. Using AND and OR, single or multiple bits can be easily cleared in one operation.

To **clear** a single bit in a port, AND the current value of the port with the value b, where $b = 255 - 2^{bit}$.

Example: Clear bit 5 in a port. Read in the current value of the port, AND it with 223 ($223 = 255 - 2^5$), and then write the resulting value to the port. In BASIC, this is programmed as:

```
V = INP(PortAddress)
V = V AND 223
OUT PortAddress, V
```

To **set** a single bit in a port, OR the current value of the port with the value b, where $b = 2^{bit}$.

Example: Set bit 3 in a port. Read in the current value of the port, OR it with 8 ($8 = 2^3$), and then write the resulting value to the port. In Pascal, this is programmed as:

```
V := Port[PortAddress];
V := V OR 8;
Port[PortAddress] := V;
```

Setting or clearing more than one bit at a time is accomplished just as easily. To **clear** multiple bits in a port, AND the current value of the port with the value b, where b = 255 - (the sum of the values of the bits to be cleared). Note that the bits do not have to be consecutive.

Example: Clear bits 2 ,4, and 6 in a port. Read in the current value of the port, AND it with 171 ($171 = 255 - 2^2 - 2^4 - 2^6$), and then write the resulting value to the port. In C, this is programmed as:

```
v = inportb(port_address);
v = v & 171;
outportb(port_address, v);
```

To **set** multiple bits in a port, OR the current value of the port with the value b, where b = the sum of the individual bits to be set. Note that the bits to be set do not have to be consecutive.

Example: Set bits 3, 5, and 7 in a port. Read in the current value of the port, OR it with 168 ($168 = 2^3 + 2^5 + 2^7$), and then write the resulting value back to the port. In assembly language, this is programmed as:

```
mov dx, PortAddress
in al, dx
or al, 168
out dx, al
```

Often, assigning a range of bits is a mixture of setting and clearing operations. You can set or clear each bit individually or use a faster method of first clearing all the bits in the range then setting only those bits that must be set using the method shown above for setting multiple bits in a port. The following example shows how this two-step operation is done.

Example: Assign bits 3, 4, and 5 in a port to 101 (bits 3 and 5 set, bit 4 cleared). First, read in the port and clear bits 3, 4, and 5 by ANDing them with 199. Then set bits 3 and 5 by ORing them with 40, and finally write the resulting value back to the port. In C, this is programmed as:

```
v = inportb(port_address);
v = v & 199;
v = v | 40;
outportb(port_address, v);
```

A final note: Don't be intimidated by the binary operators AND and OR and try to use operators for which you have a better intuition. For instance, if you are tempted to use addition and subtraction to set and clear bits in place of the methods shown above, DON'T! Addition and subtraction may seem logical, but they **will not work** if you try to clear a bit that is already clear or set a bit that is already set. For example, you might think that to set bit 5 of a port, you simply need to read in the port, add 32 (2^5) to that value, and then write the resulting value back to the port. This works fine if bit 5 is not already set. But, what happens when bit 5 *is* already set? Bits 0 to 4 will be unaffected and we can't say for sure what happens to bits 6 and 7, but we can say for sure that bit 5 ends up cleared instead of being set. A similar problem happens when you use subtraction to clear a bit in place of the method shown above.

Now that you know how to clear and set bits, we are ready to look at the programming steps for the DM5604 module functions.

D/A Conversions

D/A conversions are performed on the eight analog output channels, AOUT1 through AOUT8, by writing data to the D/A converter registers and then issuing an update command (BA + 20) which simultaneously updates the outputs of all eight channels.

The 12-bit digital data for each channel is loaded in a two step process, LSB followed by MSB. After the data has been loaded for all desired channels, the channels are simultaneously updated by issuing the update command. If a channel's data has not been updated since the last conversion, the output of the DAC will not change. The digital data is converted to a corresponding voltage which is present at the output until new data is loaded and another update command is issued.

The output voltage range of each channel is determined by the settings of the AOUT channel jumpers on P3 through P10. The following tables show key digital inputs and their corresponding outputs for unipolar and bipolar voltage ranges. The resolution for 0 to +5 volts is 1.22 millivolts; for 0 to +10 and ±5 volts, 2.44 millivolts; and for ±10 volts, 4.88 millivolts.

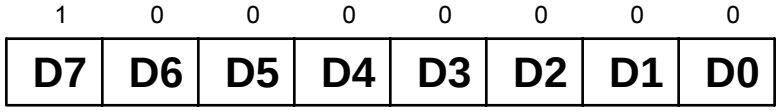
D/A Converter Unipolar Calibration Table		
	Ideal Output Voltage (millivolts)	
D/A Bit Weight	0 to +5 Volts	0 to +10 Volts
4095 (full-scale)	+4998.78	+9997.56
2048	+2500.00	+5000.00
1024	+1250.00	+2500.00
512	+625.00	+1250.00
256	+312.50	+625.00
128	+156.25	+312.50
64	+78.13	+156.25
32	+39.06	+78.13
16	+19.53	+39.06
8	+9.77	+19.53
4	+4.88	+9.77
2	+2.44	+4.88
1	+1.22	+2.44
0	0.00	0.00

D/A Converter Bipolar Calibration Table		
	Ideal Output Voltage (millivolts)	
D/A Bit Weight	-5 to +5 Volts	-10 to +10 Volts
4095 (full-scale)	+4997.56	+9995.12
2048	0000.00	0000.00
1024	-2500.00	-5000.00
512	-3750.00	-7500.00
256	-4375.00	-8750.00
128	-4687.50	-9375.00
64	-4843.75	-9687.50
32	-4921.88	-9843.75
16	-4960.94	-9921.88
8	-4980.47	-9960.94
4	-4990.23	-9980.47
2	-4995.12	-9990.23
1	-4997.56	-9995.12
0	-5000.00	-10000.00

Initializing the 8255 PPI

Before you can use the 24 8255 based digital I/O lines on your DM5604, the 8255 must be initialized. This step must be executed every time you start up, reset, or reboot your computer.

The 8255 is initialized by writing the appropriate control word to I/O port BA + 3. The contents of your control word will vary, depending on how you want to configure your I/O lines. Use the control word description in the previous I/O map section to help you program the right value. In the example below, a decimal value of 128 sets up the 8255 so that all I/O lines are Mode 0 outputs.



Digital I/O Operations

Once the 8255 is initialized, you can use the digital I/O lines to control or monitor external devices.

Interrupts

- What Is an Interrupt?

An interrupt is an event that causes the processor in your computer to temporarily halt its current process and execute another routine. Upon completion of the new routine, control is returned to the original routine at the point where its execution was interrupted.

Interrupts are very handy for dealing with asynchronous events (events that occur at less than regular intervals). Keyboard activity is a good example; your computer cannot predict when you might press a key and it would be a waste of processor time for it to do nothing while waiting for a keystroke to occur. Thus, the interrupt scheme is used and the processor proceeds with other tasks. Then, when a keystroke does occur, the keyboard 'interrupts' the processor, and the processor gets the keyboard data, places it in memory, and then returns to what it was doing before it was interrupted. Other common devices that use interrupts are modems, disk drives, and mice.

Your DM5604 can interrupt the processor when one of the three interrupt sources is enabled. By using these interrupts, you can write software that effectively deals with real world events.

- Interrupt Request Lines

To allow different peripheral devices to generate interrupts on the same computer, the PC bus has eight different interrupt request (IRQ) lines. A transition from low to high on one of these lines generates an interrupt request which is handled by the PC's interrupt controller. The interrupt controller checks to see if interrupts are to be acknowledged from that IRQ and, if another interrupt is already in progress, it decides if the new request should supersede the one in progress or if it has to wait until the one in progress is done. This prioritizing allows an interrupt to be interrupted if the second request has a higher priority. The priority level is based on the number of the IRQ; IRQ0 has the highest priority, IRQ1 is second-highest, and so on through IRQ7, which has the lowest. Many of the IRQs are used by the standard system resources. IRQ0 is used by the system timer, IRQ1 is used by the keyboard, IRQ3 by COM2, IRQ4 by COM1, and IRQ6 by the disk drives. Therefore, it is important for you to know which IRQ lines are available in your system for use by the DM5604.

- 8259 Programmable Interrupt Controller

The chip responsible for handling interrupt requests in the PC is the 8259 Programmable Interrupt Controller. To use interrupts, you will need to know how to read and set the 8259's interrupt mask register (IMR) and how to send the end-of-interrupt (EOI) command to the 8259.

- Interrupt Mask Register (IMR)

Each bit in the interrupt mask register (IMR) contains the mask status of an IRQ line; bit 0 is for IRQ0, bit 1 is for IRQ1, and so on. If a bit is **set** (equal to 1), then the corresponding IRQ is masked and it will not generate an interrupt. If a bit is **clear** (equal to 0), then the corresponding IRQ is unmasked and can generate interrupts. The IMR is programmed through port 21H.

IRQ7	IRQ6	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1	IRQ0

I/O Port 21H

For all bits:
0 = IRQ unmasked (enabled)
1 = IRQ masked (disabled)

- End-of-Interrupt (EOI) Command

After an interrupt service routine is complete, the 8259 interrupt controller must be notified. This is done by writing the value 20H to I/O port 20H.

- What Exactly Happens When an Interrupt Occurs?

Understanding the sequence of events when an interrupt is triggered is necessary to properly write software interrupt handlers. When an interrupt request line is driven high by a peripheral device (such as the DM5604), the

interrupt controller checks to see if interrupts are enabled for that IRQ, and then checks to see if other interrupts are active or requested and determines which interrupt has priority. The interrupt controller then interrupts the processor. The current code segment (CS), instruction pointer (IP), and flags are pushed on the stack for storage, and a new CS and IP are loaded from a table that exists in the lowest 1024 bytes of memory. This table is referred to as the interrupt vector table and each entry is called an interrupt vector. Once the new CS and IP are loaded from the interrupt vector table, the processor begins executing the code located at CS:IP. When the interrupt routine is completed, the CS, IP, and flags that were pushed on the stack when the interrupt occurred are now popped from the stack and execution resumes from the point where it was interrupted.

- Using Interrupts in Your Programs

Adding interrupts to your software is not as difficult as it may seem, and what they add in terms of performance is often worth the effort. Note, however, that although it is not that hard to use interrupts, the smallest mistake will often lead to a system hang that requires a reboot. This can be both frustrating and time-consuming. But, after a few tries, you'll get the bugs worked out and enjoy the benefits of properly executed interrupts.

- Writing an Interrupt Service Routine (ISR)

The first step in adding interrupts to your software is to write the interrupt service routine (ISR). This is the routine that will automatically be executed each time an interrupt request occurs on the specified IRQ. An ISR is different than standard routines that you write. First, on entrance, the processor registers should be pushed onto the stack **BEFORE** you do anything else. Second, just before exiting your ISR, you must clear the interrupt status of the DM5604 and write an end-of-interrupt command to the 8259 controller. Finally, when exiting the ISR, in addition to popping all the registers you pushed on entrance, you must use the IRET instruction and **not** a plain RET. The IRET automatically pops the flags, CS, and IP that were pushed when the interrupt was called.

If you find yourself intimidated by interrupt programming, take heart. Most Pascal and C compilers allow you to identify a procedure (function) as an interrupt type and will automatically add these instructions to your ISR, with one important exception: most compilers **do not** automatically add the end-of-interrupt command to the procedure; you must do this yourself. Other than this and the few exceptions discussed below, you can write your ISR just like any other routine. It can call other functions and procedures in your program and it can access global data. If you are writing your first ISR, we recommend that you stick to the basics; just something that will convince you that it works, such as incrementing a global variable.

NOTE: If you are writing an ISR using assembly language, you are responsible for pushing and popping registers and using IRET instead of RET.

There are a few cautions you must consider when writing your ISR. The most important is, **do not use any DOS functions or routines that call DOS functions from within an ISR**. DOS is **not** reentrant; that is, a DOS function cannot call itself. In typical programming, this will not happen because of the way DOS is written. But what about when using interrupts? Then, you could have a situation such as this in your program. If DOS function X is being executed when an interrupt occurs and the interrupt routine makes a call to DOS function X, then function X is essentially being called while it is already active. Such a reentrancy attempt spells disaster because DOS functions are not written to support it. This is a complex concept and you do not need to understand it. Just make sure that you do not call any DOS functions from within your ISR. The one wrinkle is that, unfortunately, it is not obvious which library routines included with your compiler use DOS functions. A rule of thumb is that routines which write to the screen, or check the status of or read the keyboard, and any disk I/O routines use DOS and should be avoided in your ISR.

The same problem of reentrancy exists for many floating point emulators as well, meaning you may have to avoid floating point (real) math in your ISR.

Note that the problem of reentrancy exists, no matter what programming language you are using. Even if you are writing your ISR in assembly language, DOS and many floating point emulators are not reentrant. Of course, there are ways around this problem, such as those which involve checking to see if any DOS functions are currently active when your ISR is called, but such solutions are well beyond the scope of this discussion.

The second major concern when writing your ISR is to make it as short as possible in terms of execution time. Spending long periods of time in your ISR may mean that other important interrupts are being ignored. Also, if you

spend too long in your ISR, it may be called again before you have completed handling the first run. This often leads to a hang that requires a reboot.

Your ISR should have this structure:

- Push any processor registers used in your ISR. Most C and Pascal interrupt routines automatically do this for you.
- Put the body of your routine here.
- Clear the interrupt bit on the DM5604 by writing any value to BA + 23.
- Issue the EOI command to the 8259 interrupt controller by writing 20H to port 20H.
- Pop all registers pushed on entrance. Most C and Pascal interrupt routines automatically do this for you.

The following C and Pascal examples show what the shell of your ISR should be like:

In C:

```
void interrupt ISR(void)
{
   /* Your code goes here. Do not use any DOS functions! */
   outportb(BaseAddress + 23, 0);   /* Clear DM5604 interrupt */
   outportb(0x20, 0x20);            /* Send EOI command to 8259 */
}
```

In Pascal:

```
Procedure ISR; Interrupt;
begin
   { Your code goes here. Do not use any DOS functions! }
   Port[BaseAddress + 23] := 0;     { Clear DM5604 interrupt }
   Port[$20] := $20;                { Send EOI command to 8259 }
end;
```

- Saving the Startup Interrupt Mask Register (IMR) and Interrupt Vector

The next step after writing the ISR is to save the startup state of the interrupt mask register and the interrupt vector that you will be using. The IMR is located at I/O port 21H. The interrupt vector you will be using is located in the interrupt vector table which is simply an array of 256-bit (4-byte) pointers and is located in the first 1024 bytes of memory (Segment = 0, Offset = 0). You can read this value directly, but it is a better practice to use DOS function 35H (get interrupt vector). Most C and Pascal compilers provide a library routine for reading the value of a vector. The vectors for the hardware interrupts are vectors 8 through 15, where IRQ0 uses vector 8, IRQ1 uses vector 9, and so on. Thus, if the DM5604 will be using IRQ3, you should save the value of interrupt vector 11.

Before you install your ISR, temporarily mask out the IRQ you will be using. This prevents the IRQ from requesting an interrupt while you are installing and initializing your ISR. To mask the IRQ, read in the current IMR at I/O port 21H and **set** the bit that corresponds to your IRQ (remember, setting a bit disables interrupts on that IRQ while clearing a bit enables them). The IMR is arranged so that bit 0 is for IRQ0, bit 1 is for IRQ1, and so on. See the paragraph entitled *Interrupt Mask Register (IMR)* earlier in this chapter for help in determining your IRQ's bit. After setting the bit, write the new value to I/O port 21H.

With the startup IMR saved and the interrupts on your IRQ temporarily disabled, you can assign the interrupt vector to point to your ISR. Again, you can overwrite the appropriate entry in the vector table with a direct memory write, but this is a bad practice. Instead, use either DOS function 25H (set interrupt vector) or, if your compiler provides it, the library routine for setting an interrupt vector. Remember that vector 8 is for IRQ0, vector 9 is for IRQ1, and so on.

If you need to program the source of your interrupts, do that next. For example, if you are using the programmable interval timer to generate interrupts, you must program it to run in the proper mode and at the proper rate.

Finally, clear the bit in the IMR for the IRQ you are using. This enables interrupts on the IRQ.

– Restoring the Startup IMR and Interrupt Vector

Before exiting your program, you must restore the interrupt mask register and interrupt vectors to the state they were in when your program started. To restore the IMR, write the value that was saved when your program started to I/O port 21H. Restore the interrupt vector that was saved at startup with either DOS function 35H (get interrupt vector), or use the library routine supplied with your compiler. Performing these two steps will guarantee that the interrupt status of your computer is the same after running your program as it was before your program started running.

- Common Interrupt Mistakes

- Remember that hardware interrupts are numbered 8 through 15, even though the corresponding IRQs are numbered 0 through 7.
- Two of the most common mistakes when writing an ISR are forgetting to clear the interrupt status of the DM5604 and forgetting to issue the EOI command to the 8259 interrupt controller before exiting the ISR.

Example Programs

Included with the DM5604 is a set of example programs that demonstrate the use of many of the module's features. These examples are in written in C, Pascal, and BASIC. Also included is an easy-to-use menu-driven diagnostics program, 5604DIAG, which is especially helpful when you are first checking out your module after installation.

Before using the software included with your module, make a backup copy of the disk. You may make as many backups as you need.

C and Pascal Programs

These programs are source code files so that you can easily develop your own custom software for your DM5604 module.

DAC	Simple program that shows how to program the D/A converters.
DIGITAL	Simple program that shows how to read and write the digital I/O lines.

BASIC Programs

These programs are source code files so that you can easily develop your own custom software for your DM5604 module.

DAC	Simple program that shows how to program the D/A converters.
DIGITAL	Simple program that shows how to read and write the digital I/O lines.

CHAPTER 5

CALIBRATION

This chapter tells you how to calibrate the DM5604 using the 5604DIAG calibration program included in the example software package and eight trimpots on the module. These trimpots calibrate the D/A X2 multiplier output.

This chapter tells you how to calibrate the D/A converter X2 voltage multiplier. The X1 range does not have to be calibrated. All D/A ranges are factory-calibrated before shipping. Any time you suspect inaccurate readings, you can check the accuracy of your conversions using the procedure below, and make adjustments as necessary. Using the 5604DIAG diagnostics program is a convenient way to monitor conversions while you calibrate the module.

Calibration is done with the module installed in your system. You can access the trimpots along the top edge of the module. Power up the system and let the board circuitry stabilize for 15 minutes before you start calibrating.

Required Equipment

The following equipment is required for calibration:

- Precision Voltage Source: -10 to +10 volts
- Digital Multimeter: 5-1/2 digits
- Small Screwdriver (for trimpot adjustment)

While not required, the 5604DIAG diagnostics program (included with example software) is helpful when performing calibrations. Figure 5-1 shows the module layout with the eight trimpots located along the top edge of the board (TR1 through TR8, left to right).

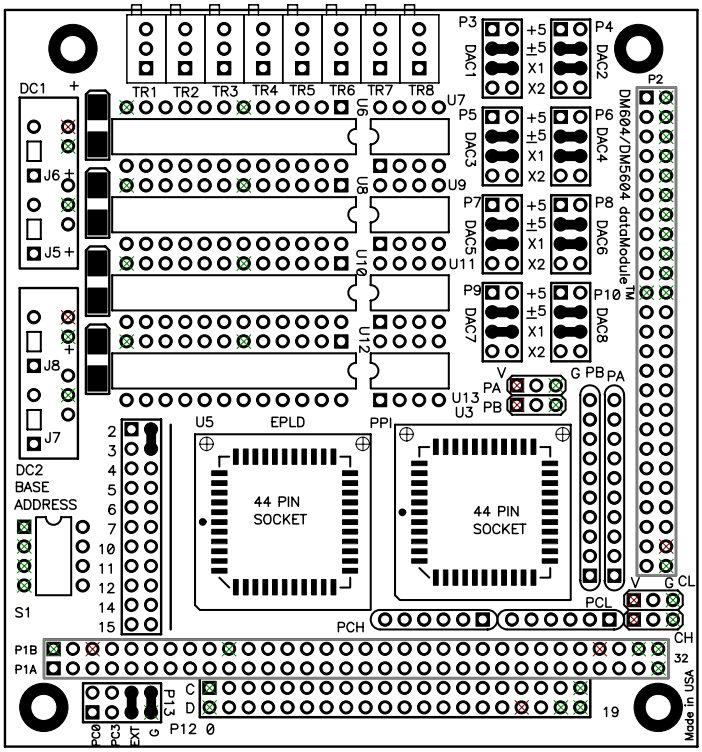


Fig. 5-1 — Module Layout

D/A Calibration — X2 Voltage Multiplier

The D/A converter requires no calibration for the X1 ranges (0 to +5 and ±5 volts). The following paragraph describes the calibration procedure for the X2 multiplier ranges.

To calibrate for X2 (0 to +10 or ±10 volts), set the DAC output voltage range to 0 to +10 volts (jumpers on 5V and X2 on the corresponding header connector which configures the DAC output for the channel you are calibrating). Then, program the D/A converter of the channel you are calibrating with the digital value 2048. The ideal DAC output for 2048 at X2 (0 to +10 volt range) is 5.0000 volts. Adjust the appropriate trimpot as listed in Table 5-1 until an output of 5.0000 volts is obtained. Repeat this procedure for all channels. Table 5-2 list the ideal output voltages for all bit weights in the unipolar ranges, and Table 5-3 lists the ideal output voltages for the bipolar ranges.

Table 5-1: X2 Voltage Adjustment	
Channel 1	TR1
Channel 2	TR2
Channel 3	TR3
Channel 4	TR4
Channel 5	TR5
Channel 6	TR6
Channel 7	TR7
Channel 8	TR8

D/A Converter Unipolar Calibration Table		
	Ideal Output Voltage (millivolts)	
D/A Bit Weight	0 to +5 Volts	0 to +10 Volts
4095 (full-scale)	+4998.78	+9997.56
2048	+2500.00	+5000.00
1024	+1250.00	+2500.00
512	+625.00	+1250.00
256	+312.50	+625.00
128	+156.25	+312.50
64	+78.13	+156.25
32	+39.06	+78.13
16	+19.53	+39.06
8	+9.77	+19.53
4	+4.88	+9.77
2	+2.44	+4.88
1	+1.22	+2.44
0	0.00	0.00

D/A Converter Bipolar Calibration Table		
	Ideal Output Voltage (millivolts)	
D/A Bit Weight	-5 to +5 Volts	-10 to +10 Volts
4095 (full-scale)	+4997.56	+9995.12
2048	0000.00	0000.00
1024	-2500.00	-5000.00
512	-3750.00	-7500.00
256	-4375.00	-8750.00
128	-4687.50	-9375.00
64	-4843.75	-9687.50
32	-4921.88	-9843.75
16	-4960.94	-9921.88
8	-4980.47	-9960.94
4	-4990.23	-9980.47
2	-4995.12	-9990.23
1	-4997.56	-9995.12
0	-5000.00	-10000.00

APPENDIX A

DM5604 SPECIFICATIONS

DM5604 Characteristics Typical @ 25° C

Interface

Switch-selectable base address, I/O mapped
Jumper-selectable interrupts

D/A Converter AD7237

Analog outputs ..... 8 channels
Resolution ..... 12 bits
Output ranges ..... 0 to +5, ±5, 0 to +10, or ±10 volts
Relative accuracy ..... ±1 LSB, max
Full-scale accuracy ..... ±5 LSB, max
Non-linearity ..... ±1 LSB, max
Settling time ..... 5 µsec, typ

Digital I/O ..... CMOS 82C55

Number of lines 24
Logic compatibility ..... TTL/CMOS
(Configurable with optional I/O pull-up/pull-down resistors)
High-level output voltage ..... 4.2V, min
Low-level output voltage ..... 0.45V, max
High-level input voltage ..... 2.2V, min; 5.5V, max
Low-level input voltage ..... -0.3V, min; 0.8V, max
Input load current ..... ±10 µA
Input capacitance,
C(IN)@F=1MHz ..... 10 pF
Output capacitance,
C(OUT)<@F=1MHz ..... 20 pF

Current Requirements

470 mA @ +5 volts (2.35W)

Connector

50-pin right angle header

Environmental

Operating temperature ..... 0 to +70°C
Storage temperature ..... -40 to +85°C
Humidity ..... 0 to 90% non-condensing

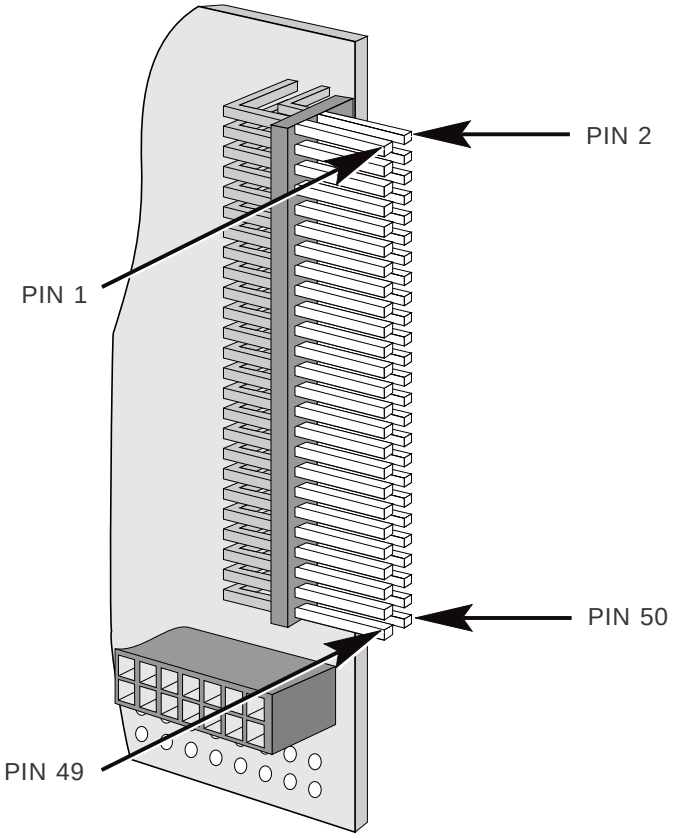
Size

3.55"L x 3.775"W x 0.6"H (90mm x 96mm x 16mm)

APPENDIX B

P2 CONNECTOR PIN ASSIGNMENTS

AOUT1	1	2	ANALOG GND
AOUT2	3	4	ANALOG GND
AOUT3	5	6	ANALOG GND
AOUT4	7	8	ANALOG GND
AOUT5	9	10	ANALOG GND
AOUT6	11	12	ANALOG GND
AOUT7	13	14	ANALOG GND
AOUT8	15	16	ANALOG GND
N.C.	17	18	ANALOG GND
EXTINT	19	20	ANALOG GND
DIGITAL GND	21	22	ANALOG GND
PA7	23	24	PC7
PA6	25	26	PC6
PA5	27	28	PC5
PA4	29	30	PC4
PA3	31	32	PC3
PA2	33	34	PC2
PA1	35	36	PC1
PA0	37	38	PC0
PB7	39	40	PB6
PB5	41	42	PB4
PB3	43	44	PB2
PB1	45	46	PB0
+12 VOLTS	47	48	+5 VOLTS
-12 VOLTS	49	50	DIGITAL GND



P2 Mating Connector Part Numbers	
Manufacturer	**Part Number**
AMP	1-746094-0
3M	3425-7650

APPENDIX C

COMPONENT DATA SHEETS

Intel 82C55A Programmable Peripheral Interface
Data Sheet Reprint

APPENDIX D WARRANTY AND RETURN POLICY

Return Policy

If you wish to return a product to the factory for service, please follow this procedure:

Read the Limited Warranty to familiarize yourself with our warranty policy.

Contact the factory for a Return Merchandise Authorization (RMA) number.

Please have the following available:

- Complete board name
- Board serial number
- A detailed description of the board's behavior

List the name of a contact person, familiar with technical details of the problem or situation, **along with their phone and fax numbers, address, and e-mail address** (if available).

List your shipping address!!

Indicate the shipping method you would like used to return the product to you.
We will not ship by next-day service without your pre-approval.

Carefully package the product, using proper anti-static packaging.

Write the RMA number in large (1") letters on the outside of the package.

Return the package to:

RTD Embedded Technologies, Inc.

103 Innovation Blvd.

State College PA 16803-0906
USA

LIMITED WARRANTY

RTD Embedded Technologies, Inc. warrants the hardware and software products it manufactures and produces to be free from defects in materials and workmanship for one year following the date of shipment from RTD Embedded Technologies, INC. This warranty is limited to the original purchaser of product and is not transferable.

During the one year warranty period, RTD Embedded Technologies will repair or replace, at its option, any defective products or parts at no additional charge, provided that the product is returned, shipping prepaid, to RTD Embedded Technologies. All replaced parts and products become the property of RTD Embedded Technologies. Before returning any product for repair, customers are required to contact the factory for an RMA number.

THIS LIMITED WARRANTY DOES NOT EXTEND TO ANY PRODUCTS WHICH HAVE BEEN DAMAGED AS A RESULT OF ACCIDENT, MISUSE, ABUSE (such as: use of incorrect input voltages, improper or insufficient ventilation, failure to follow the operating instructions that are provided by RTD Embedded Technologies, "acts of God" or other contingencies beyond the control of RTD Embedded Technologies), OR AS A RESULT OF SERVICE OR MODIFICATION BY ANYONE OTHER THAN RTD Embedded Technologies. EXCEPT AS EXPRESSLY SET FORTH ABOVE, NO OTHER WARRANTIES ARE EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND RTD Embedded Technologies EXPRESSLY DISCLAIMS ALL WARRANTIES NOT STATED HEREIN. ALL IMPLIED WARRANTIES, INCLUDING IMPLIED WARRANTIES FOR MECHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED TO THE DURATION OF THIS WARRANTY. IN THE EVENT THE PRODUCT IS NOT FREE FROM DEFECTS AS WARRANTED ABOVE, THE PURCHASER'S SOLE REMEDY SHALL BE REPAIR OR REPLACEMENT AS PROVIDED ABOVE. UNDER NO CIRCUMSTANCES WILL RTD Embedded Technologies BE LIABLE TO THE PURCHASER OR ANY USER FOR ANY DAMAGES, INCLUDING ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES, EXPENSES, LOST PROFITS, LOST SAVINGS, OR OTHER DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PRODUCT.

SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES FOR CONSUMER PRODUCTS, AND SOME STATES DO NOT ALLOW LIMITATIONS ON HOW LONG AN IMPLIED WARRANTY LASTS, SO THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE.

RTD Embedded Technologies, Inc.
103 Innovation Blvd.
State College PA 16803-0906
USA
Our website: www.rtd.com

DM5604 Board User-Selected Settings	
Base I/O Address:	
(hex)	(decimal)
Interrupts:	
EXT	IRQ Channel:
PC0	IRQ Channel:
PC3	IRQ Channel: